河南大學

Henan University

I'm dying to play GTA 6

Contestant

范恒嘉 **Hengjia Fan**

宋明贤 **Mingxian Song**

张帆 **Fan Zhang**

目录

1 图论

1.1 2-SAT

1.1.1 2-SAT

```cpp
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

const int N = 2e6+10;

vector<int> gra[N];
int dfn[N];//时间戳，节点x第一次被访问的顺序
int low[N];//追溯值，从节点x出发，所能访问到的最早时间戳
int tot;//时间戳编号
int stk[N];//栈
int instk[N];//在不在栈中
int top;
int scc[N];//点x属于哪一个scc
int siz[N];//scc大小
int cnt;//scc编号

void tarjan(int x){
    //入x时，盖戳，入栈
    dfn[x] = low[x] = ++ tot;
    stk[++top] =  x;
    instk[x] = 1;

    for(int y : gra[x]){
        if(!dfn[y]){
            tarjan(y);
            low[x] = min(low[x],low[y]);
        }else if(instk[y]){
            low[x] = min(low[x],dfn[y]);
        }
    }

    if(dfn[x] == low[x]){
        int y;
        ++cnt;
        do{
            y = stk[top--];
            instk[y] = 0;
            scc[y] = cnt;
            ++siz[cnt];
        }while(y!=x);
    }
}

void solve(){
    int n,m;
    cin>>n>>m;

    for(int i = 1;i<=m;i++){
        int a,va,b,vb;
        cin>>a>>va>>b>>vb;


        gra[a + !va*n].push_back(b+vb*n);
        gra[b + !vb*n].push_back(a+va*n);
    }


    for(int i = 1;i<=n*2;i++){
        if(!dfn[i]) tarjan(i);
    }

    // for(int i = 1;i<=n;i++){
    //     cout<<scc[i]<<' '<<scc[i+n]<<'\n';

    // }
    for(int i = 1;i<=n;i++){
        if(scc[i] == scc[i+n]){
            cout<<"IMPOSSIBLE";
            return ;
        }
    }

    cout<<"POSSIBLE"<<'\n';

    for(int i = 1;i<=n;i++){
        cout<<(scc[i] > scc[i+n])<<' ';
    }

```

```
}

int main()
{
    Song4u

    int Test_number = 1;
//    cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.2 DSU

```
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

struct DSU
{
    vector<int> f, siz;

    DSU(int n) : f(n), siz(n, 1)
    {
        iota(f.begin(), f.end(), 0);
    }

    int find(int x)
    {
        while (f[x] != x)
            x = f[x] = f[f[x]];
        return x;
    }

    bool merge(int x, int y)
    {
        x = find(x);
        y = find(y);

        if (x == y)
            return false;

        if (siz[x] < siz[y])
            swap(x, y);

        siz[x] += siz[y];
        f[y] = x;
        return true;
    }

    int size(int x)
    {
        return siz[find(x)];
    }

    bool connected(int x, int y)
    {
        return find(x) == find(y);
    }
};



void solve(){
    int n;
    cin>>n;

    DSU dsu(n+1);

    vector<pii> a(n+1);

    for(int i = 1;i<=n;i++) cin>>a[i].fi>>a[i].se;

    int ans = 1;

    for(int i = 1;i<=n;i++){
       auto [l,r] = a[i];
       l--;
       if(dsu.find(l) == dsu.find(r)){
        ans = 0;
        break;
       }else{
        dsu.merge(l,r);
       }
    }
```

```
    cout<<ans<<'\n';
}

int main()
{
    Song4u

    int Test_number = 1;
    cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.3 LCA

1.3.1 最近公共祖先

```
// //倍增版
// const int N = 5e5+10;
// int n,m,s;
// vector<int> gra[N];
// int dep[N],fa[N][20];

// void dfs(int u,int pa){
//     dep[u] = dep[pa] + 1;

//     fa[u][0] = pa;
//     for(int i = 1;i<=19;i++){
//         fa[u][i] = fa[fa[u][i-1]][i-1];
//     }

//     for(auto x : gra[u]){
//         if(x!=pa) dfs(x,u);
//     }
// }

// int lca(int u,int v){
//     if(dep[u]<dep[v]) swap(u,v);

//     for(int i = 19;i>=0;i--){
//         if(dep[fa[u][i]] >= dep[v]) u = fa[u][i];
//     }

//     if(u==v) return v;

//     for(int i = 19;i>=0;i--){
//         if(fa[u][i] != fa[v][i]){
//             u = fa[u][i];
//             v = fa[v][i];
//         }
//     }

//     return fa[u][0];
// }

// //树链剖分版
// const int N = 5e5+10;

// vector<vector<int>> gra(N);

// int son[N];
// int fa[N];
// int dep[N];
// int sz[N];
// int top[N];

// int n,m,s;

// void dfs(int u,int pa){
//     dep[u] = dep[pa] + 1;
//     fa[u] = pa;
//     sz[u] = 1;


//     for(auto x : gra[u]){
//         if(x==pa) continue;
//         dfs(x,u);
//         sz[u] += sz[x];
//         if(sz[son[u]] < sz[x]) son[u] = x;
//     }
// }

// void gettop(int u,int t){
//     top[u] = t;

//     if(!son[u]) return ;

//     gettop(son[u],t);

//     for(auto x : gra[u]){
//         if(x==son[u] || x==fa[u]) continue;
//         gettop(x,x);
//     }
// }
```

```
78
79 // int lca(int u,int v){
80 //     while(top[u] != top[v]){
81 //         if(dep[top[u]] < dep[top[v]]) swap(u,v);
82 //         u = fa[top[u]];
83 //     }
84
85 //     return dep[u] < dep[v] ? u : v;
86 // }
```

1.4 二分图

1.4.1 all

```
1  //染色法判二分图
2  // const int N = 1e5+10;
3  // const int M = 2*N;
4
5  // int n,m;
6  // struct edge{
7  //     int v,ne;
8  // }e[M];
9
10 // int h[N];
11 // int idx;
12 // int col[N];
13
14 // void add(int a,int b){
15 //     e[idx++] = {b,h[a]};
16 //     h[a] = idx;
17 // }
18
19 // bool dfs(int u,int c){
20 //     col[u] = c;
21 //     for(int i = h[u];i;i = e[i].ne){
22 //         int v = e[i].v;
23 //         if(!col[v]){
24 //             if(dfs(v,3-c)) return true;
25 //         }else{
26 //             if(col[v] == c) return false;
27 //         }
28 //     }
29
30 //     return false;
31 // }
32
33 // void solve(){
34 //     cin>>n>>m;
35
36 //     for(int i = 1;i<=m;i++){
37 //         int a,b;
38 //         cin>>a>>b;
39
40 //         add(a,b);
41 //         add(b,a);
42 //     }
43
44 //     bool suc = true;
45 //     for(int i = 1;i<=n;i++){
46 //         if(!col[i]){
47 //             if(dfs(i,1)){
48 //                 suc = false;
49 //                 break;
50 //             }
51 //         }
52 //     }
53
54 //     cout<<(suc ? "YES" : "NO")<<'\n';
55 // }
56
57
58 //二分图最大匹配
59 // std::mt19937_64 rng(
60 //     static_cast<std::mt19937_64::result_type>(std::time(nullptr)));
61 // struct BipartiteGraph {
62 //   int n1, n2;                       // number of vertices in X and Y, resp.
63 //   std::vector<std::vector<int>> g;  // edges from X to Y
64 //   std::vector<int> ma, mb;  // matches from X to Y and from Y to X, resp.
65 //   std::vector<bool> vis;    // visiting marks for DFS.
66
67 //   BipartiteGraph(int n1, int n2)
68 //       : n1(n1), n2(n2), g(n1), ma(n1, -1), mb(n2, -1) {}
69
70 //   // Add an edge from u in X to v in Y.
71 //   void add_edge(int u, int v) { g[u].emplace_back(v); }
72
73 //   // Find an augmenting path starting at u.
74 //   bool dfs(int u) {
75 //     vis[u] = true;
76 //     // Heuristic: find unsaturated vertices whenever possible.
77 //     for (int v : g[u]) {
78 //       if (mb[v] == -1) {
79 //         ma[u] = v;
80 //         mb[v] = u;
81 //         return true;
82 //       }
```

```
//     }
//     for (int v : g[u]) {
//       if (!vis[mb[v]] && dfs(mb[v])) {
//         ma[u] = v;
//         mb[v] = u;
//         return true;
//       }
//     }
//     return false;
//   }

//   // Kuhn's maximum matching algorithm.
//   std::vector<std::pair<int, int>> kuhn_maximum_matching() {
//     // Randomly shuffle the edges.
//     for (int u = 0; u < n1; ++u) {
//       std::shuffle(g[u].begin(), g[u].end(), rng);
//     }
//     // Find a maximal set of vertex-disjoint augmenting paths in each round.
//     while (true) {
//       bool succ = false;
//       vis.assign(n1, false);
//       for (int u = 0; u < n1; ++u) {
//         succ |= ma[u] == -1 && dfs(u);
//       }
//       if (!succ) break;
//     }
//     // Collect the matched pairs.
//     std::vector<std::pair<int, int>> matches;
//     matches.reserve(n1);
//     for (int u = 0; u < n1; ++u) {
//       if (ma[u] != -1) {
//         matches.emplace_back(u, ma[u]);
//       }
//     }
//     return matches;
//   }
// };

// void solve(){
//     int n,m,e;
//     cin>>n>>m>>e;

//     BipartiteGraph bg(n+1,m+1);

//     for(int i = 1;i<=e;i++){
//         int u,v;
//         cin>>u>>v;
//         bg.add_edge(u,v);
//     }

//     vector<pii> q = bg.kuhn_maximum_matching();

//     cout<<q.size()<<'\n';

// }

//二分图最大权值匹配
// template <typename T>
// struct hungarian {  // km
//   int n;
//   vector<int> matchx;  // 左集合对应的匹配点
//   vector<int> matchy;  // 右集合对应的匹配点r
//   vector<int> pre;     // 连接右集合的左点
//   vector<bool> visx;   // 拜访数组 左
//   vector<bool> visy;   // 拜访数组 右
//   vector<T> lx;
//   vector<T> ly;
//   vector<vector<T>> g;
//   vector<T> slack;
//   T inf;
//   T res;
//   queue<int> q;
//   int org_n;
//   int org_m;

//   hungarian(int _n, int _m) {
//     org_n = _n;
//     org_m = _m;
//     n = max(_n, _m);
//     inf = numeric_limits<T>::max();
//     res = 0;
//     g = vector<vector<T>>(n, vector<T>(n));
//     matchx = vector<int>(n, -1);
//     matchy = vector<int>(n, -1);
//     pre = vector<int>(n);
//     visx = vector<bool>(n);
//     visy = vector<bool>(n);
//     lx = vector<T>(n, -inf);
//     ly = vector<T>(n);
//     slack = vector<T>(n);
//   }

//   void addEdge(int u, int v, int w) {
//     g[u][v] = max(w, 0);  // 负值还不如不匹配 因此设为0不影响
//   }

```

```
//   bool check(int v) {
//     visy[v] = true;
//     if (matchy[v] != -1) {
//       q.push(matchy[v]);
//       visx[matchy[v]] = true;  // in S
//       return false;
//     }
//     // 找到新的未匹配点 更新匹配点 pre 数组记录着"非匹配边"上与之相连的点
//     while (v != -1) {
//       matchy[v] = pre[v];
//       swap(v, matchx[pre[v]]);
//     }
//     return true;
//   }

//   void bfs(int i) {
//     while (!q.empty()) {
//       q.pop();
//     }
//     q.push(i);
//     visx[i] = true;
//     while (true) {
//       while (!q.empty()) {
//         int u = q.front();
//         q.pop();
//         for (int v = 0; v < n; v++) {
//           if (!visy[v]) {
//             T delta = lx[u] + ly[v] - g[u][v];
//             if (slack[v] >= delta) {
//               pre[v] = u;
//               if (delta) {
//                 slack[v] = delta;
//               } else if (check(v)) {  // delta=0 代表有机会加入相等子图 找增广路
//                                       // 找到就return 重建交错树
//                 return;
//               }
//             }
//           }
//         }
//       }
//       // 没有增广路 修改顶标
//       T a = inf;
//       for (int j = 0; j < n; j++) {
//         if (!visy[j]) {
//           a = min(a, slack[j]);
//         }
//       }
//       for (int j = 0; j < n; j++) {
//         if (visx[j]) {  // S
//           lx[j] -= a;
//         }
//         if (visy[j]) {  // T
//           ly[j] += a;
//         } else {  // T'
//           slack[j] -= a;
//         }
//       }
//       for (int j = 0; j < n; j++) {
//         if (!visy[j] && slack[j] == 0 && check(j)) {
//           return;
//         }
//       }
//     }
//   }

//   void solve() {
//     // 初始顶标
//     for (int i = 0; i < n; i++) {
//       for (int j = 0; j < n; j++) {
//         lx[i] = max(lx[i], g[i][j]);
//       }
//     }

//     for (int i = 0; i < n; i++) {
//       fill(slack.begin(), slack.end(), inf);
//       fill(visx.begin(), visx.end(), false);
//       fill(visy.begin(), visy.end(), false);
//       bfs(i);
//     }

//     // custom
//     for (int i = 0; i < n; i++) {
//       if (g[i][matchx[i]] > 0) {
//         res += g[i][matchx[i]];
//       } else {
//         matchx[i] = -1;
//       }
//     }
//     cout << res << "\n";
//     for (int i = 0; i < org_n; i++) {
//       cout << matchx[i] + 1 << " ";
//     }
//     cout << "\n";
//   }
// };
```

1.4.2 二分图最大匹配

```cpp
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

std::mt19937_64 rng(
    static_cast<std::mt19937_64::result_type>(std::time(nullptr)));
struct BipartiteGraph {
  int n1, n2;                       // number of vertices in X and Y, resp.
  std::vector<std::vector<int>> g;  // edges from X to Y
  std::vector<int> ma, mb;  // matches from X to Y and from Y to X, resp.
  std::vector<bool> vis;    // visiting marks for DFS.

  BipartiteGraph(int n1, int n2)
      : n1(n1), n2(n2), g(n1), ma(n1, -1), mb(n2, -1) {}

  // Add an edge from u in X to v in Y.
  void add_edge(int u, int v) { g[u].emplace_back(v); }

  // Find an augmenting path starting at u.
  bool dfs(int u) {
    vis[u] = true;
    // Heuristic: find unsaturated vertices whenever possible.
    for (int v : g[u]) {
      if (mb[v] == -1) {
        ma[u] = v;
        mb[v] = u;
        return true;
      }
    }
    for (int v : g[u]) {
      if (!vis[mb[v]] && dfs(mb[v])) {
        ma[u] = v;
        mb[v] = u;
        return true;
      }
    }
    return false;
  }

  // Kuhn's maximum matching algorithm.
  std::vector<std::pair<int, int>> kuhn_maximum_matching() {
    // Randomly shuffle the edges.
    for (int u = 0; u < n1; ++u) {
      std::shuffle(g[u].begin(), g[u].end(), rng);
    }
    // Find a maximal set of vertex-disjoint augmenting paths in each round.
    while (true) {
      bool succ = false;
      vis.assign(n1, false);
      for (int u = 0; u < n1; ++u) {
        succ |= ma[u] == -1 && dfs(u);
      }
      if (!succ) break;
    }
    // Collect the matched pairs.
    std::vector<std::pair<int, int>> matches;
    matches.reserve(n1);
    for (int u = 0; u < n1; ++u) {
      if (ma[u] != -1) {
        matches.emplace_back(u, ma[u]);
      }
    }
    return matches;
  }
};

void solve(){
    int n,m,e;
    cin>>n>>m>>e;

    BipartiteGraph bg(n+1,m+1);

    for(int i = 1;i<=e;i++){
        int u,v;
        cin>>u>>v;
        bg.add_edge(u,v);
    }

    vector<pii> q = bg.kuhn_maximum_matching();

    cout<<q.size()<<'\n';

}

```

```cpp
int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.4.3 二分图最大权值匹配

```cpp
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
// #define inf 0x3f3f3f3f
// #define infll 0x3f3f3f3f3f3f3f3fLL

template <typename T>
struct hungarian {  // km
  int n;
  vector<int> matchx;  // 左集合对应的匹配点
  vector<int> matchy;  // 右集合对应的匹配点
  vector<int> pre;     // 连接右集合的左点
  vector<bool> visx;   // 拜访数组 左
  vector<bool> visy;   // 拜访数组 右
  vector<T> lx;
  vector<T> ly;
  vector<vector<T>> g;
  vector<T> slack;
  T inf;
  T res;
  queue<int> q;
  int org_n;
  int org_m;

  hungarian(int _n, int _m) {
    org_n = _n;
    org_m = _m;
    n = max(_n, _m);
    inf = numeric_limits<T>::max();
    res = 0;
    g = vector<vector<T>>(n, vector<T>(n));
    matchx = vector<int>(n, -1);
    matchy = vector<int>(n, -1);
    pre = vector<int>(n);
    visx = vector<bool>(n);
    visy = vector<bool>(n);
    lx = vector<T>(n, -inf);
    ly = vector<T>(n);
    slack = vector<T>(n);
  }

  void addEdge(int u, int v, int w) {
    g[u][v] = max(w, 0);  // 负值还不如不匹配 因此设为0不影响
  }

  bool check(int v) {
    visy[v] = true;
    if (matchy[v] != -1) {
      q.push(matchy[v]);
      visx[matchy[v]] = true;  // in S
      return false;
    }
    // 找到新的未匹配点 更新匹配点 pre 数组记录着"非匹配边"上与之相连的点
    while (v != -1) {
      matchy[v] = pre[v];
      swap(v, matchx[pre[v]]);
    }
    return true;
  }

  void bfs(int i) {
    while (!q.empty()) {
      q.pop();
    }
    q.push(i);
    visx[i] = true;
    while (true) {
      while (!q.empty()) {
        int u = q.front();
        q.pop();
        for (int v = 0; v < n; v++) {
          if (!visy[v]) {
            T delta = lx[u] + ly[v] - g[u][v];
            if (slack[v] >= delta) {
              pre[v] = u;
```

```
            if (delta) {
              slack[v] = delta;
            } else if (check(v)) {  // delta=0 代表有机会加入相等子图 找增广路
                                    // 找到就return 重建交错树
              return;
            }
          }
        }
      }
      // 没有增广路 修改顶标
      T a = inf;
      for (int j = 0; j < n; j++) {
        if (!visy[j]) {
          a = min(a, slack[j]);
        }
      }
      for (int j = 0; j < n; j++) {
        if (visx[j]) {  // S
          lx[j] -= a;
        }
        if (visy[j]) {  // T
          ly[j] += a;
        } else {  // T'
          slack[j] -= a;
        }
      }
      for (int j = 0; j < n; j++) {
        if (!visy[j] && slack[j] == 0 && check(j)) {
          return;
        }
      }
    }
  }

  int solve() {
    // 初始顶标
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
        lx[i] = max(lx[i], g[i][j]);
      }
    }

    for (int i = 0; i < n; i++) {
      fill(slack.begin(), slack.end(), inf);
      fill(visx.begin(), visx.end(), false);
      fill(visy.begin(), visy.end(), false);
      bfs(i);
    }

    // custom
    for (int i = 0; i < n; i++) {
      if (g[i][matchx[i]] > 0) {
        res += g[i][matchx[i]];
      } else {
        matchx[i] = -1;
      }
    }
    return res;

    // for (int i = 0; i < org_n; i++) {
    //   cout << matchx[i] + 1 << " ";
    // }
    // cout << "\n";
  }
};



void solve() {
    int n;
    cin>>n;

    hungarian<int> h(n+1,n+1);
    hungarian<int> g(n+1,n+1);

    vector<vector<int>> mp(n+1,vector<int> (n+1));

    for(int i = 1;i<=n;i++){
        for(int j = 1;j<=n;j++){
            cin>>mp[i][j];

            h.addEdge(i,j,mp[i][j]);
            g.addEdge(i,j,100-mp[i][j]);
        }
    }

    cout<<n*100-g.solve()<<'\n';
    cout<<h.solve();



}

int main() {
```

```
    Song4u
    int Test_number = 1;
    // cin >> Test_number;
    while(Test_number--) solve();
    return 0;
}
```

1.4.4 染色法

```
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

const int N = 1e5+10;
const int M = 2*N;

int n,m;
struct edge{
    int v,ne;
}e[M];

int h[N];
int idx;
int col[N];

void add(int a,int b){
    e[idx++] = {b,h[a]};
    h[a] = idx;
}

bool dfs(int u,int c){
    col[u] = c;
    for(int i = h[u];i;i = e[i].ne){
        int v = e[i].v;
        if(!col[v]){
            if(dfs(v,3-c)) return true;
        }else{
            if(col[v] == c) return false;
        }
    }

    return false;
}

void solve(){
    cin>>n>>m;

    for(int i = 1;i<=m;i++){
        int a,b;
        cin>>a>>b;

        add(a,b);
        add(b,a);
    }

    bool suc = true;
    for(int i = 1;i<=n;i++){
        if(!col[i]){
            if(dfs(i,1)){
                suc = false;
                break;
            }
        }
    }

    cout<<(suc ? "YES" : "NO")<<'\n';
}

int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.5 最小生成树

1.5.1 最小生成树

```
// //克鲁斯卡尔
// //每个点最初是一个连通块，每次选取边权最小的边，当边连接的是两个不同的连通块，把
```

```
    他们合并成一个连通块，直至生成树中有n个点
3
4  // const int N = 2e5 + 3;
5  // const int M = 2e6 + 3;
6  // struct kruskal
7  // {
8  //     struct Edge
9  //     {
10 //         int u, v, w;
11 //         bool operator<(const Edge &f) const
12 //         {
13 //             return w < f.w;
14 //         }
15 //     };
16 //     Edge e[M];
17 //     int tot,n;
18 //     int f[N];
19 //     int find(int x)
20 //     {
21 //         return f[x] == -1 ? x : (f[x] = find(f[x]));
22 //     }
23 //     bool merge(int u,int v)
24 //     {
25 //         u=find(u),v=find(v);
26 //         if(u==v)
27 //             return false;
28 //         f[u]=v;
29 //         return true;
30 //     }
31 //     void init(int nn)
32 //     {
33 //         n = nn;
34 //         tot = 0;
35 //         memset(f, -1, sizeof(int) * (n + 1));
36 //     }
37 //     void addEdge(int u, int v, int w)
38 //     {
39 //         e[tot++] = {u, v, w};
40 //     }
41 //     int run()
42 //     {
43 //         sort(e, e + tot);
44 //         int cnt = 1,cost = 0;
45 //         for (int i = 0;i<tot && cnt<n;++i)
46 //         {
47 //             if(!merge(e[i].u,e[i].v))
48 //                 continue;
49 //             cost += e[i].w;
50 //             ++cnt;
51 //         }
52 //         return (cnt==n?cost:-1);
53 //     }
54 // };
55
56
57 // //prim
58 // //任意点作为生成树的起点，每次选取和生成树中的点距离最小的非生成树中的点，加入生
       成树，直至生成树中有n个点
59
60 // const int N = 5e3 + 3;
61 // struct prim
62 // {
63 //     struct Edge//注意顺序，因为优先队列是小根堆，之后同理
64 //     {
65 //         int w,v;
66 //         bool operator>(const Edge &f)const
67 //         {
68 //             return w > f.w;
69 //         }
70 //     };
71 //     vector<Edge> e[N];
72 //     priority_queue<Edge,vector<Edge>,greater<Edge> > q;
73 //     int dis[N],vis[N];
74 //     int n;
75 //     void init(int nn)
76 //     {
77 //         n = nn;
78 //         for(int i=1;i<=n;++i)
79 //             e[i].clear();
80 //         memset(dis,0x3f,sizeof(int)*(n+1));
81 //         memset(vis,0,sizeof(int)*(n+1));
82 //         while(!q.empty())
83 //             q.pop();
84 //     }
85 //     void addEdge(int u, int v, int w)
86 //     {
87 //         e[u].push_back({w,v});
88 //         e[v].push_back({w,u});
89 //     }
90 //     int run()
91 //     {
92 //         dis[1]=0;
93 //         q.push({0,1});
94 //         Edge u;
95 //         int cnt=0,cost=0;
96 //         while(!q.empty() && cnt<n)
```

```
//         {
//             u=q.top();
//             q.pop();
//             if(vis[u.v])
//                 continue;
//             vis[u.v]=1;
//             cost+=u.w;
//             ++cnt;
//             for(auto &[w,v]:e[u.v])
//             {
//                 if(w>=dis[v])
//                     continue;
//                 dis[v]=w;
//                 q.push({w,v});
//             }
//         }
//         return (cnt==n?cost:-1);
//     }
// };


// //Boruvka
// //每次处理一个连通块时，它向其他连通块伸出一条最小边权的边，启发式合并两个连通块的点（这里总共花费 nlogn）
// //如果能连成，就加上这个边的贡献，直至连通块数为1。

// int a[N];//点权，1-index
// namespace boruvka
// {
//     int n,f[N];
//     vector<int> connectBlock[N];//连通块 i 包含的点权(或点)
//     queue<int> q;//待处理的点（连通块）
//     int find(int x)
//     {
//         return f[x] ==0 ? x : (f[x] = find(f[x]));
//     }
//     bool merge(int u,int v)//启发式合并维护连通块包含的点
//     {
//         u=find(u),v=find(v);
//         if(u==v)
//             return false;
//         if(connectBlock[u].size()<connectBlock[v].size())
//             swap(u,v);
//         for(int &x:connectBlock[v])
//             connectBlock[u].push_back(x);
//         connectBlock[v].clear();
//         f[v]=u;
//         return true;
//     }
//     void init(int nn)
//     {
//         n = nn;
//         for(int i=1;i<=n;++i)
//             q.push(i);
//         for(int i=1;i<=n;++i)
//             connectBlock[i].push_back(a[i]);
//     }
//     array<int,2> cal(int x)
//     {
//         //根据题目计算最小边权和对应的点
//         return {0,0};
//     }
//     ll run()
//     {
//         array<int,2> res;//边权，连接的另一个点（连通块），顺序不能错
//         int u,com=n;//连通块数
//         ll ans=0;
//         while(com>1)
//         {
//             while(connectBlock[q.front()].empty())
//                 q.pop();
//             u=q.front();
//             q.pop();
//             res={INF,0};
//             for(int &v:connectBlock[u])
//                 res=min(res,cal(v));
//             if(!merge(u,res[1]))
//                 continue;
//             q.push(u);
//             ans+=res[0];
//             --com;
//         }
//         return ans;
//     }
// };
```

1.6 最短路

1.6.1 Bellman ford

```cpp
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
```

```cpp
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

const int N = 1e5+10;
int n;
vector<pii> gra[N];

bool bellmanford(int st){ //bellmanford判负环
    vector<ll> dist(n+1,infll);
    dist[st] = 0;

    bool suc = false;

    for(int i = 1;i<=n;i++){
        suc = false;

        for(int u = 1;u<=n;u++){
            for(auto [v,len] : gra[u]){
                if(dist[u]==infll) continue;
                if(dist[v] > dist[u] + len){
                    dist[v] = dist[u] + len;
                    suc = true;

                }
            }
        }

        if(!suc) break;

    }

    return suc;

}

void solve(){

}

int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.6.2 Floyd

```cpp
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

const int N = 1005; // 假设最大顶点数

struct Floyd {
    int n;
    ll dis[N][N];

    void init(int m) {
        n = m;
        for (int i = 0; i <= n; ++i) {
            for (int j = 0; j <= n; ++j) {
                dis[i][j] = (i==j) ? 0 : infll;
            }
        }
    }

    void flo() {
        for (int k = 1; k <= n; ++k) {
            for (int i = 1; i <= n; ++i) {
                for (int j = 1; j <= n; ++j) {
                    if (dis[i][j] > dis[i][k] + dis[k][j]) {
                        dis[i][j] = dis[i][k] + dis[k][j];
                    }
                }
```

```
            }
        }
    }
};

void solve(){
    int n,m;
    cin>>n>>m;


    Floyd f;
    f.init(n);

    for(int i = 1;i<=m;i++){
        int u,v,w;
        cin>>u>>v>>w;

        f.dis[u][v] = min(f.dis[u][v],w+0ll);
        f.dis[v][u] = min(f.dis[v][u],w+0ll);
    }


    f.flo();

    for(int i = 1;i<=n;i++){
        for(int j = 1;j<=n;j++){
            cout<<f.dis[i][j] << ' ';
        }
        cout<<endl;
    }
}

int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.6.3 all

```
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define T double
#define ptt pair<double,double>
#define tri tuple<int,int,int>
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

vector<vector<vector<ll>>> floyd(vector<vector<vector<ll>>> dist,int n){
    for (int k = 1; k <= n; k++) {
        for (int x = 1; x <= n; x++) {
            for (int y = 1; y <= n; y++) {
                dist[k][x][y] = min(dist[k - 1][x][y], dist[k - 1][x][k] + dist[k - 1][k][y]);
            }
        }
    }

    return dist;
}

vector<ll> dijkstra(vector<vector<pll>> gra,int n,int st){//不能处理边权为负的情况
    ll ans = 0;
    priority_queue<pll,vector<pll>,greater<pll>> q;

    vector<ll> dist(n+1,infll);
    vector<bool> vis(n+1,false);


    q.push({0,st});
    dist[st] = 0;

    while(!q.empty()){
        auto [len,tnod] = q.top();
        q.pop();

        if(vis[tnod]) continue;
        vis[tnod] = true;

        for(auto [nnod,w] : gra[tnod]){
            if(dist[nnod] > dist[tnod] + w){
                dist[nnod] = dist[tnod] + w;
                q.push({dist[nnod],nnod});
            }
        }
```

```
    }

    return dist;
}

bool bellmanford(vector<vector<pll>> gra,int n,int st){ //bellmanford判负环
    vector<ll> dist(n+1,infll);
    dist[st] = 0;

    bool suc = false;

    for(int i = 1;i<=n;i++){
        suc = false;

        for(int u = 1;u<=n;u++){
            for(auto [v,len] : gra[u]){
                if(dist[u]==infll) continue;
                if(dist[v] > dist[u] + len){
                    dist[v] = dist[u] + len;
                    suc = true;

                }
            }
        }

        if(!suc) break;

    }

    return suc;

}

bool spfa(vector<vector<pll>> gra,int n,int st){//判负环
    vector<ll> dist(n+1,infll);
    vector<bool> vis(n+1,false);
    vector<int> cnt(n+1,0);//记录到达该点最短路径的经过边数
                           //若这个数大于n则一定经过负环
                           //因为最多只可能经过n-1条边

    dist[st] = 0;
    vis[st] = true;

    queue<int> q;
    q.push(st);

    while(!q.empty()){
        int tnod = q.front();
        q.pop();
        vis[tnod] = false;
        for(auto [nnod,w] : gra[tnod]){
            if(dist[nnod] > dist[tnod] + w){
                dist[nnod] = dist[tnod] + w;
                cnt[tnod]  = cnt[nnod] + 1;

                if(cnt[nnod] >= n) return false;
                if(!vis[nnod]) q.push(nnod);
                vis[nnod] = true;
            }
        }
    }

    return true;
}


void solve(){

}

int main()
{
    Song4u

    int Test_number = 1;
//    cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.6.4 dijikstra

```
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
```

```cpp
#define infll 0x3f3f3f3f3f3f3f3fLL

const int N = 1e5+10;
vector<pii> gra[N];
int n;
vector<ll> dijkstra(int st){//不能处理边权为负的情况
    ll ans = 0;
    priority_queue<pll,vector<pll>,greater<pll>> q;

    vector<ll> dist(n+1,infll);
    vector<bool> vis(n+1,false);


    q.push({0,st});
    dist[st] = 0;

    while(!q.empty()){
        auto [len,tnod] = q.top();
        q.pop();

        if(vis[tnod]) continue;
        vis[tnod] = true;

        for(auto [nnod,w] : gra[tnod]){
            if(dist[nnod] > dist[tnod] + w){
                dist[nnod] = dist[tnod] + w;
                q.push({dist[nnod],nnod});
            }
        }
    }

    return dist;
}

void solve(){
    cin>>n;
}

int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.6.5 spfa

```cpp
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

const int N = 1e5+10;
int n;

vector<pii> gra[N];
bool spfa(vector<vector<pll>> gra,int n,int st){//判负环
    vector<ll> dist(n+1,infll);
    vector<bool> vis(n+1,false);
    vector<int> cnt(n+1,0);//记录到达该点最短路径的经过边数
                           //若这个数大于n则一定经过负环
                           //因为最多只可能经过n-1条边

    dist[st] = 0;
    vis[st] = true;

    queue<int> q;
    q.push(st);

    while(!q.empty()){
        int tnod = q.front();
        q.pop();
        vis[tnod] = false;
        for(auto [nnod,w] : gra[tnod]){
            if(dist[nnod] > dist[tnod] + w){
                dist[nnod] = dist[tnod] + w;
                cnt[tnod]  = cnt[nnod] + 1;

                if(cnt[nnod] >= n) return false;
                if(!vis[nnod]) q.push(nnod);
                vis[nnod] = true;
            }
        }
    }
```

```

    return true;
}


void solve(){
    cin>>n;
}

int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.7 连通性（tarjan）

1.7.1 all

```
// tarjan求强连通分量
// e.g. https://www.luogu.com.cn/problem/P2863
// const int N = 1e4+10;

// vector<int> gra[N];
// int dfn[N];//时间戳，节点x第一次被访问的顺序
// int low[N];//追溯值，从节点x出发，所能访问到的最早时间戳
// int tot;//时间戳编号
// int stk[N];//栈
// int instk[N];//在不在栈中
// int top;
// int scc[N];//点x属于哪一个scc
// int siz[N];//scc大小
// int cnt;//scc编号

// void tarjan(int x){
//     //入x时，盖戳，入栈
//     dfn[x] = low[x] = ++ tot;
//     stk[++top] =  x;
//     instk[x] = 1;

//     for(int y : gra[x]){
//         if(!dfn[y]){
//             tarjan(y);
//             low[x] = min(low[x],low[y]);
//         }else if(instk[y]){
//             low[x] = min(low[x],dfn[y]);
//         }
//     }

//     if(dfn[x] == low[x]){
//         int y;
//         ++cnt;
//         do{
//             y = stk[top--];
//             instk[y] = 0;
//             scc[y] = cnt;
//             ++siz[cnt];
//         }while(y!=x);
//     }
// }

//tarjan scc 缩点
//e.g.https://www.luogu.com.cn/problem/P2341
//e.g.https://www.luogu.com.cn/problem/P3387
//e.g.https://www.luogu.com.cn/problem/P3387
// vector<int> gra[N];
// int dfn[N];//时间戳，节点x第一次被访问的顺序
// int low[N];//追溯值，从节点x出发，所能访问到的最早时间戳
// int tot;//时间戳编号
// int stk[N];//栈
// int instk[N];//在不在栈中
// int top;
// int scc[N];//点x属于哪一个scc
// int siz[N];//scc大小
// int cnt;//scc编号
// int din[N],dout[N];//统计scc的入度和出度

// void tarjan(int x){
//     //入x时，盖戳，入栈
//     dfn[x] = low[x] = ++ tot;
//     stk[++top] =  x;
//     instk[x] = 1;

//     for(int y : gra[x]){
//         if(!dfn[y]){
//             tarjan(y);
//             low[x] = min(low[x],low[y]);
//         }else if(instk[y]){
//             low[x] = min(low[x],dfn[y]);
//         }
//     }
```

```
//     if(dfn[x] == low[x]){
//         int y;
//         ++cnt;
//         do{
//             y = stk[top--];
//             instk[y] = 0;
//             scc[y] = cnt;
//             ++siz[cnt];
//         }while(y!=x);
//     }
// }

//  for(int i = 1;i<=n;i++){
//         for(int j : gra[i]){
//             if(scc[i] != scc[j]){
//                 din[scc[j]] ++;
//                 dout[scc[i]] ++;
//             }
//         }
//     }



//tarjan判割点
//e.g.https://www.luogu.com.cn/problem/P3388

// const int N = 2e4+10;

// vector<int> gra[N];
// int dfn[N];//时间戳，节点x第一次被访问的顺序
// int low[N];//追溯值，从节点x出发，所能访问到的最早时间戳
// int tot;//时间戳编号
// int cut[N];//记录当前点是不是割点
// int root;//哪个点是根


// void tarjan(int x){
//     //入x时，盖戳
//     dfn[x] = low[x] = ++tot;
//     int child = 0;

//     for(int y : gra[x]){
//         if(!dfn[y]){//若y尚未访问
//             tarjan(y);
//             //回x时，更新low，判割点
//             low[x] = min(low[x],low[y]);
//             if(low[y] >= dfn[x]){
//                 child ++;//子树的数量
//                 if(x != root || child > 1){
//                     cut[x] = 1;
//                 }
//             }
//         }else{//若y已经找到
//             low[x] = min(low[x],dfn[y]);
//         }

//     }
// }
// void solve(){
//     cin>>n>>m;

//     while(m--){
//         int u,v;
//         cin>>u>>v;

//         gra[u].push_back(v);
//         gra[v].push_back(u);
//     }


//     for(int i = 1;i<=n;i++){
//         if(!dfn[i]){
//             root  = i;
//             tarjan(i);
//         }
//     }

//     int ans = 0;
//     for(int i = 1;i<=n;i++){
//         if(cut[i]) ans++;
//     }

//     cout<<ans<<'\n';

//     for(int i = 1;i<=n;i++) {
//         if(cut[i]) cout<<i<<' ';
//     }
// }




//tarjan找割边
//e.g.https://www.luogu.com.cn/problem/P1656

```

```cpp
// const int N=210,M=10010;
// int n,m,a,b;

// struct edge{int u,v;};
// vector<edge>e;//边集
// vector<int>h[N];//出边
// int dfn[N],low[N],tot;

// struct bridge{
//   int x,y;
//   bool operator<(const bridge &t)const{
//     if(x==t.x)return y<t.y;
//     return x<t.x;
//   }
// }bri[M];
// int cnt;//给割边计数

// void add(int a,int b){
//   e.push_back({a,b});
//   h[a].push_back(e.size()-1);
// }
// void tarjan(int x,int in_edge){
//   dfn[x]=low[x]=++tot;
//   for(int i=0;i<h[x].size();i++){
//     int j=h[x][i], y=e[j].v;//j是当前出边在边集中的位置，y是当前边的下一个点
//     if(!dfn[y]){ //若y尚未访问
//       tarjan(y,j);
//       low[x]=min(low[x],low[y]);
//       if(low[y]>dfn[x]){
//         bri[cnt++]={x,y};
//       }
//     }
//     else if(j!=(in_edge^1)) //不是反边
//       low[x]=min(low[x],dfn[y]);
//   }
// }


// void solve(){
//     cin>>n>>m;

//     while(m--){
//         cin>>a>>b;

//         add(a,b);
//         add(b,a);
//     }

//     for(int i = 1;i<=n;i++){
//         if(!dfn[i]) tarjan(i,0);
//     }

//     sort(bri,bri+cnt);

//     for(int i = 0;i<cnt;i++){
//         cout<<bri[i].x<<' '<<bri[i].y<<'\n';
//     }
// }



// //eDCC：无向图中极大的不包含割边的连通块被称为“边双连通分量”

//适用于无向有环图得到森林或树
// const int M = 1e4+10;
// const int N = 5e3+10;

// int n,m;
// struct edge{
//     int v,ne;
// };
// edge e[M];
// int h[N],idx = 1;
// int dfn[N];
// int low[N];
// int tot;
// stack<int> stk;
// int dcc[N],cnt;
// int bri[M];
// int d[N];

// void add(int x,int y){
//     e[++idx].v = y;
//     e[idx].ne = h[x];
//     h[x] = idx;
// }

// void tarjan(int x,int in_edg){
//     dfn[x] = low[x] = ++tot;
//     stk.push(x);

//     for(int i = h[x];i;i = e[i].ne){
//         int y = e[i].v;

//         if(!dfn[y]){
//             tarjan(y,i);
```

```
//             low[x] = min(low[x],low[y]);
//             if(low[y] > dfn[x]){
//                 bri[i] = bri[i^1] = true;
//             }
//         }else if(i!=(in_edg^1)){
//             low[x] = min(low[x],dfn[y]);
//         }
//     }

//     if(dfn[x] == low[x]){
//         ++cnt;

//         while(1){
//             int y = stk.top();
//             stk.pop();
//             dcc[y] = cnt;
//             if(y==x) break;
//         }
//     }
// }

// void solve(){
//     cin>>n>>m;

//     while(m--){
//         int x,y;
//         cin>>x>>y;

//         add(x,y);
//         add(y,x);
//     }

//     tarjan(1,0);

//     for(int i = 2;i<=idx;i++){
//         if(bri[i]){
//             d[dcc[e[i].v]] ++;
//         }
//     }

//     int ans = 0;

//     for(int i = 1;i<=cnt;i++){
//         if(d[i] == 1) ans++;
//     }

//     cout<<ans<<'\n';
//     ans = (ans+1) / 2;


//     cout<<ans;

// }


// // tarjan求vDCC
// //e.g.https://www.luogu.com.cn/problem/P8435
// const int N = 1e4+10;
// int n,m;
// vector<int> e[N];//加边数组
// vector<int> ne[N];//new_e在缩完点之后要建的新图
// int dfn[N];
// int low[N];
// int tot;
// stack<int> stk;
// vector<int> dcc[N];//点双连通分量
// int cnt;//记录dcc
// int cut[N];//点x是不是割点
// int root;
// int num;//给每个割点一个新编号
// int id[N];//给每个割点新编号用

// void tarjan(int x){
//     dfn[x] = low[x] = ++ tot;
//     stk.push(x);
//     if(!e[x].size()){//若x没有边，就是一个孤立点
//         dcc[++cnt].push_back(x);
//         return ;
//     }

//     int child = 0;
//     for(int y : e[x]){
//         if(!dfn[y]){
//             tarjan(y);
//             low[x] = min(low[x],low[y]);
//             if(low[y] >= dfn[x]){
//                 child ++;
//                 if(x!=root || child > 1){
//                     cut[x] = true;
//                 }

//                 cnt++;

//                 while(1){
//                     int z = stk.top();
//                     stk.pop();
//                     dcc[cnt].push_back(z);
```

```
//                      if(z==y) break;
//                  }
//
//                  dcc[cnt].push_back(x);
//              }
//          }else{
//              low[x] = min(low[x],dfn[y]);
//          }
//      }
// }

// void solve(){
//      cin>>n>>m;

//      while(m--){
//          int a,b;
//          cin>>a>>b;

//          e[a].push_back(b);
//          e[b].push_back(a);
//      }

//      for(root = 1;root<=n;root ++){
//          if(!dfn[root]) tarjan(root);
//      }

//      //给每个割点一个新编号，因为缩点之后要把割点裂点
//      num = cnt;
//      for(int i = 1;i<=n;i++){
//          if(cut[i]) id[i] = ++num;
//      }

//      //建新图，从每个vdcc向对应割点连边
//      for(int i = 1;i<=cnt;i++){
//          for(int j = 0;j<dcc[i].size();j++){
//              int x = dcc[i][j];
//              if(cut[x]){
//                  ne[i].push_back(id[x]);
//                  ne[id[x]].push_back(i);
//              }
//          }
//      }
// }
```

1.7.2 tarjan SCC 缩点

```
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define T double
#define ptt pair<double,double>
#define tri tuple<int,int,int>
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

int n,m;


const int N = 1e4+10;

vector<int> gra[N];
int dfn[N];//时间戳，节点x第一次被访问的顺序
int low[N];//追溯值，从节点x出发，所能访问到的最早时间戳
int tot;//时间戳编号
int stk[N];//栈
int instk[N];//在不在栈中
int top;
int scc[N];//点x属于哪一个scc
int siz[N];//scc大小
int cnt;//scc编号
int din[N],dout[N];//统计scc的入度和出度

void tarjan(int x){
    //入x时，盖戳，入栈
    dfn[x] = low[x] = ++ tot;
    stk[++top] =  x;
    instk[x] = 1;

    for(int y : gra[x]){
        if(!dfn[y]){
            tarjan(y);
            low[x] = min(low[x],low[y]);
        }else if(instk[y]){
            low[x] = min(low[x],dfn[y]);
        }
    }

    if(dfn[x] == low[x]){
        int y;
        ++cnt;
        do{
```

```
            y = stk[top--];
            instk[y] = 0;
            scc[y] = cnt;
            ++siz[cnt];
        }while(y!=x);
    }
}



void solve(){
    cin>>n>>m;

    for(int i = 1;i<=m;i++){
        int u,v;
        cin>>u>>v;

        gra[u].push_back(v);
    }

    for(int i = 1;i<=n;i++){
        if(!dfn[i]){
            tarjan(i);
        }
    }

    if(cnt==1){
        cout<<n;
        return ;
    }
    for(int i = 1;i<=n;i++){
        for(int j : gra[i]){
            if(scc[i] != scc[j]){
                din[scc[j]] ++;
                dout[scc[i]] ++;
            }
        }
    }

    int ans = 0;
    int z = 0;
    for(int i = 1;i<=cnt;i++){
        if(dout[i] == 0){
            ans+=siz[i];
            z++;

        }
    }

    cout<<(z > 1 ? 0 : ans);
}

int main()
{
    Song4u

    int Test_number = 1;
//    cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.7.3 tarjan 求 eDCC

```
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define T double
#define ptt pair<double,double>
#define tri tuple<int,int,int>
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

//eDCC：无向图中极大的不包含割边的连通块被称为“边双连通分量”
//e.g.https://www.luogu.com.cn/problem/P2860
const int M = 1e4+10;
const int N = 5e3+10;

int n,m;
struct edge{
    int v,ne;
};
edge e[M];
int h[N],idx = 1;
int dfn[N];
int low[N];
int tot;
stack<int> stk;
int dcc[N],cnt;
int bri[M];
```

```
int d[N];

void add(int x,int y){
    e[++idx].v = y;
    e[idx].ne = h[x];
    h[x] = idx;
}

void tarjan(int x,int in_edg){
    dfn[x] = low[x] = ++tot;
    stk.push(x);

    for(int i = h[x];i;i = e[i].ne){
        int y = e[i].v;

        if(!dfn[y]){
            tarjan(y,i);
            low[x] = min(low[x],low[y]);
            if(low[y] > dfn[x]){
                bri[i] = bri[i^1] = true;
            }
        }else if(i!=(in_edg^1)){
            low[x] = min(low[x],dfn[y]);
        }
    }

    if(dfn[x] == low[x]){
        ++cnt;

        while(1){
            int y = stk.top();
            stk.pop();
            dcc[y] = cnt;
            if(y==x) break;
        }
    }
}

void solve(){
    cin>>n>>m;

    while(m--){
        int x,y;
        cin>>x>>y;

        add(x,y);
        add(y,x);
    }

    tarjan(1,0);

    for(int i = 2;i<=idx;i++){
        if(bri[i]){
            d[dcc[e[i].v]] ++;
        }
    }

    int ans = 0;

    for(int i = 1;i<=cnt;i++){
        if(d[i] == 1) ans++;
    }

    // cout<<ans<<'\n';
    ans = (ans+1) / 2;

    cout<<ans;

}

int main()
{
    Song4u

    int Test_number = 1;
//    cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.7.4 tarjan 求 vDCC

1.7.5 tarjan 求割点

```
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define T double
#define ptt pair<double,double>
```

```cpp
#define tri tuple<int,int,int>
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

int n,m;


const int N = 2e4+10;

vector<int> gra[N];
int dfn[N];//时间戳，节点x第一次被访问的顺序
int low[N];//追溯值，从节点x出发，所能访问到的最早时间戳
int tot;//时间戳编号
int cut[N];//记录当前点是不是割点
int root;//哪个点是根


void tarjan(int x){
    //入x时，盖戳
    dfn[x] = low[x] = ++tot;
    int child = 0;

    for(int y : gra[x]){
        if(!dfn[y]){//若y尚未访问
            tarjan(y);
            //回x时，更新low，判割点
            low[x] = min(low[x],low[y]);
            if(low[y] >= dfn[x]){
                child ++;//子树的数量
                if(x != root || child > 1){
                    cut[x] = 1;
                }
            }
        }else{//若y已经找到
            low[x] = min(low[x],dfn[y]);
        }

    }
}



void solve(){
    cin>>n>>m;

    while(m--){
        int u,v;
        cin>>u>>v;

        gra[u].push_back(v);
        gra[v].push_back(u);
    }


    for(int i = 1;i<=n;i++){
        if(!dfn[i]){
            root  = i;
            tarjan(i);
        }
    }

    int ans = 0;
    for(int i = 1;i<=n;i++){
        if(cut[i]) ans++;
    }

    cout<<ans<<'\n';

    for(int i = 1;i<=n;i++) {
        if(cut[i]) cout<<i<<' ';
    }
}

int main()
{
    Song4u

    int Test_number = 1;
//    cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.7.6 tarjan 求割边

```cpp
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
```

```cpp
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

const int N=210,M=10010;
int n,m,a,b;

struct edge{int u,v;};
vector<edge>e;//边集
vector<int>h[N];//出边
int dfn[N],low[N],tot;

struct bridge{
  int x,y;
  bool operator<(const bridge &t)const{
    if(x==t.x)return y<t.y;
    return x<t.x;
  }
}bri[M];
int cnt;//给割边计数

void add(int a,int b){
  e.push_back({a,b});
  h[a].push_back(e.size()-1);
}
void tarjan(int x,int in_edge){
  dfn[x]=low[x]=++tot;
  for(int i=0;i<h[x].size();i++){
    int j=h[x][i], y=e[j].v;//j是当前出边在边集中的位置，y是当前边的下一个点
    if(!dfn[y]){ //若y尚未访问
      tarjan(y,j);
      low[x]=min(low[x],low[y]);
      if(low[y]>dfn[x]){
        bri[cnt++]={x,y};
      }
    }
    else if(j!=(in_edge^1)) //不是反边
      low[x]=min(low[x],dfn[y]);
  }
}


void solve(){
    cin>>n>>m;

    while(m--){
        cin>>a>>b;

        add(a,b);
        add(b,a);
    }

    for(int i = 1;i<=n;i++){
        if(!dfn[i]) tarjan(i,0);
    }

    sort(bri,bri+cnt);

    for(int i = 0;i<cnt;i++){
        cout<<bri[i].x<<' '<<bri[i].y<<'\n';
    }
}

int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

1.7.7 tarjan 求强连通分量

```cpp
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL


// https://www.luogu.com.cn/problem/P2863
const int N = 1e4+10;

```

```cpp
int n,m;
vector<int> gra[N];
int dfn[N];//时间戳，节点x第一次被访问的顺序
int low[N];//追溯值，从节点x出发，所能访问到的最早时间戳
int tot;//时间戳编号
int stk[N];//栈
int instk[N];//在不在栈中
int top;
int scc[N];//点x属于哪一个scc
int siz[N];//scc大小
int cnt;//scc编号

void tarjan(int x){
    //入x时，盖戳，入栈
    dfn[x] = low[x] = ++ tot;
    stk[++top] =  x;
    instk[x] = 1;

    for(int y : gra[x]){
        if(!dfn[y]){
            tarjan(y);
            low[x] = min(low[x],low[y]);
        }else if(instk[y]){
            low[x] = min(low[x],dfn[y]);
        }
    }

    if(dfn[x] == low[x]){
        int y;
        ++cnt;
        do{
            y = stk[top--];
            instk[y] = 0;
            scc[y] = cnt;
            ++siz[cnt];
        }while(y!=x);
    }
}

void solve(){
    cin>>n>>m;
    while(m--){
        int a,b;
        cin>>a>>b;

        gra[a].push_back(b);
    }

    for(int i = 1;i<=n;i++){
        if(!dfn[i]) tarjan(i);
    }

    int ans = 0;
    for(int i = 1;i<=cnt;i++){
        if(siz[i] > 1) ans++;
    }
    cout<<ans;
}

int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

2 二分 & 三分

```cpp
//O.o?  bug again??????
#include <bits/stdc++.h>
using namespace std;

#define Song4u ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define ll long long
#define pii pair<int,int>
#define pll pair<long long,long long>
#define pdd pair<double,double>
#define tri tuple<int,int,int>
#define fi first
#define se second
#define inf 0x3f3f3f3f
#define infll 0x3f3f3f3f3f3f3f3fLL

const double EPS = 1e-9;

void solve(){

    int l,r,mid;
    int ans;
    auto check = [&](int mid) -> bool{

```

```
    };
    //整数
    while(l<=r)
    {
        mid=(l+r)/2;
        if(check(mid))
        {
            l=mid+1;
            ans=mid;
        }
        else
            r=mid-1;
    }

    //实数
    int times=100;
    while(times--)
    {
        mid=(l+r)/2;
        if(check(mid))
        {
            l=mid;
            ans=mid;
        }
        else
            r=mid;
    }

        //整数
    while(l<=r)
    {
        mid=(l+r)/2;
        if(check(mid)<check(mid+1))
        {
            l=mid+1;
            ans=mid;
        }
        else
            r=mid-1;
    }

    //实数
    int times=100;
    while(times--)
    {
        mid=(l+r)/2;
        if(check(mid)<check(mid+EPS))
        {
            l=mid;
            ans=mid;
        }
        else
            r=mid;
    }
}

int main()
{
    Song4u

    int Test_number = 1;
    // cin>>Test_number;
    while(Test_number--) solve();
    return 0;

}
```

3 凸包

```
#include <bits/stdc++.h>
using i64 = long long;

constexpr long double EPS = 1E-10;

template<typename T>
int sgn(T a){
    return (a > EPS) - (a < -EPS);
}

template<typename T>
struct Point;

template<typename T>
struct Vector { //向量
    T x = 0, y = 0;
    Vector(const Point<T> &p) : x(p.x), y(p.y) {}
    Vector(T x_ = 0, T y_ = 0) : x(x_), y(y_) {}
    template<typename U> operator Vector<U>() {return Vector<U>(U(x), U(y));}
    Vector operator+(const Vector<T> &o) const {return {x + o.x, y + o.y};}
    Vector operator-(const Vector<T> &o) const {return {x - o.x, y - o.y};}
    Vector operator*(T f) const {return {x * f, y * f};}
    Vector operator/(T f) const {return {x / f, y / f};}
    friend std::istream &operator>>(std::istream &is, Vector &p) {
        return is >> p.x >> p.y;
    }
```

```
    friend std::ostream &operator<<(std::ostream &os, Vector p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }
};

template<typename T>
struct Point { //点
    T x = 0, y = 0;
    Point(T x_ = 0, T y_ = 0) : x(x_), y(y_) {}
    template<typename U> operator Point<U>() {return Point<U>(U(x), U(y));}
    Point operator+(const Vector<T> &o) const {return {x + o.x, y + o.y};}
    Vector<T> operator-(const Point<T> &o) const {return {x - o.x, y - o.y};}
    Point operator-() const {return {-x, -y};}
    Point operator*(T f) const {return {x * f, y * f};}
    Point operator/(T f) const {return {x / f, y / f};}
    friend Point operator*(T f, Point p) {return {p * f};}
    bool operator==(const Point &o) const {
        return sgn(x - o.x) == 0 && sgn(y - o.y) == 0;
    }
    constexpr std::strong_ordering operator<=>(const Point &o) const {
        if(sgn(x - o.x) == 0) {
            return sgn(y - o.y) <=> 0;
        } else {
            return sgn(x - o.x) <=> 0;
        }
    }
    friend std::istream &operator>>(std::istream &is, Point &p) {
        return is >> p.x >> p.y;
    }
    friend std::ostream &operator<<(std::ostream &os, Point p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }
};

template<typename T>
struct Line { //直线
    Point<T> s, t;
    Line() = default;
    Line(Point<T> _s, Point<T> _t) : s(_s), t(_t) {}
};

template<typename T>
struct Seg { //线段
    Point<T> s, t;
    Seg() = default;
    Seg(Point<T> _s, Point<T> _t) : s(_s), t(_t) {}
};

template<typename T, typename U>
struct Circle { //圆
    Point<T> o;
    U r{};
};

template<typename T>
T dot(const Vector<T> &a, const Vector<T> &b) { //向量a和向量b的点积
    return a.x * b.x + a.y * b.y;
}

template<typename T>
T dot(const Point<T> &a, const Point<T> &b, const Point<T> &c) { //向量a->b和向量a->c的点积
    return dot(b - a, c - a);
}

template<typename T>
T cross(const Vector<T> &a, const Vector<T> &b) { //向量a和向量b的叉积
    return a.x * b.y - a.y * b.x;
}

template<typename T>
T cross(const Point<T> &a, const Point<T> &b, const Point<T> &c) { //向量a->b和向量a->c的叉积
    return cross(b - a, c - a);
}

template<typename T>
T len2(const Vector<T> &a) { //向量a的模长的平方
    return a.x * a.x + a.y * a.y;
}

template<typename T>
long double len(const Vector<T> &a) { //向量a的模长
    return sqrtl(len2(a));
}

template<typename T>
Vector<T> standardize(const Vector <T> &a) { //向量a的单位向量
    return a / len(a);
}

template<typename T>
long double angle(Vector<T> a, Vector<T> b) {//求两向量夹角[0, pi]
    return fabs(atan2l(cross(a, b), dot(a, b)));
}
```

```cpp
template<typename T>
T dis2(const Point<T> &a, const Point<T> &b) { //点a和点b距离的平方（防精度损失）
    return (b.x - a.x) * (b.x - a.x) + (b.y - a.y) * (b.y - a.y);
}

template<typename T>
long double dis(const Point<T> &a, const Point<T> &b) { //点a到点b距离
    return sqrtl(dis2(a, b));
}

template<typename T>
long double angle(const Vector<T> &a, const Vector<T> &b) { //向量a和向量b的夹角弧度
    return acosl(dot(a, b) / len(a) / len(b));
}

template<typename T>
Vector<T> rotate(const Vector <T> &a) { //向量a逆时针旋转pi/2
    return {-a.y, a.x};
}

template<typename T>
Vector<T> rotate(const Vector <T> &a, long double rad) { //向量a逆时针旋转rad弧度
    return {a.x * cosl(rad) - a.y * sinl(rad), a.x * sinl(rad) + a.y * cosl(rad)};
}

template<typename T>
Point<T> rotate(const Point<T> &a, const Point<T> &b, long double rad) { //点b绕点a逆时针旋转rad弧度
    return {
        (b.x - a.x) * cosl(rad) - (b.y - a.y) * sinl(rad) + a.x,
        (b.x - a.x) * sinl(rad) + (b.y - a.y) * cosl(rad) + a.y
    };
}

template<typename T>
bool intersect(const Seg<T> &a, const Seg<T> &b) { //线段a和线段b不严格相交，可以包含端点相交
    return sgn(cross(a.s, a.t, b.s) * cross(a.s, a.t, b.t)) <= 0
        && sgn(cross(b.s, b.t, a.s) * cross(b.s, b.t, a.t)) <= 0;
}

template<typename T>
bool intersectStrictly(const Seg<T> &a, const Seg<T> &b) { //线段a和线段b严格相交，不包含端点相交
    return sgn(cross(a.s, a.t, b.s) * cross(a.s, a.t, b.t)) < 0
        && sgn(cross(b.s, b.t, a.s) * cross(b.s, b.t, a.t)) < 0;
}

template<typename T>
Point<T> getNode(const Line<T> &a, const Line<T> &b) { //求直线a与直线b的交点（需要先判断是否相交）
    T dx = cross(a.s, b.s, b.t) / cross(a.t - a.s, b.t - b.s);
    return a.s + (a.t - a.s) * dx;
}

template<typename T>
Point<T> getNode(const Seg<T> &a, const Seg<T> &b) { //求线段a与线段b的交点（需要先判断是否相交）
    T dx = cross(a.s, b.s, b.t) / cross(a.t - a.s, b.t - b.s);
    return a.s + (a.t - a.s) * dx;
}

template<typename T>
Line<T> midseg(const Seg<T> &seg) { //求线段的垂直平分线
    Point mid = (seg.s + seg.t) / 2;
    return {mid, mid + rotate(seg.t - seg.s)};
}

template<typename T>
Circle<T, long double> getCircle(const Point<T> &p1, const Point<T> &p2, const Point<T> &p3) { //求三点所在圆
    Line<T> l1 = midseg(Seg<T>{p1, p2});
    Line<T> l2 = midseg(Seg<T>{p1, p3});
    Point<T> O = getNode(l1, l2);
    return {O, dis(O, p1)};
}

template<typename T>
Circle<T, long double> getCircle(const Point<T> &p1, const Point<T> &p2) { //求以两点为直径所在圆
    Point<T> O = (p1 + p2) / 2;
    return {O, dis(O, p1)};
}

template<typename T>
bool onSeg(const Point<T> &p, const Seg<T> &s) { //判断点是否在线段上
    if(sgn(cross(s.s, s.t, p)) != 0) return false;
    return std::min(s.s, s.t) <= p && p <= std::max(s.s, s.t);
}

template<typename T, typename U>
int pointOnCircle(const Point<T> &p, const Circle<T, U> &c) { //点和圆的位置关系，-1点在圆外，0点在圆上，1点在圆外，
    return sgn(c.r * c.r - dis2(p, c.o));
}
```

```cpp
template<typename T>
std::vector<Point<T>> andrew(std::vector<Point<T>> v) { //Andrew求凸包
    std::sort(v.begin(), v.end());
    std::vector<Point<T>> stk;
    for(int i = 0; i < v.size(); ++i) {
        while(stk.size() > 1 && sgn(cross(stk[stk.size() - 2], stk.back(), v[i])) <= 0) {
            stk.pop_back();
        }
        stk.push_back(v[i]);
    }
    int t = stk.size();
    for(int i = (int)v.size() - 2; i >= 0; --i) {
        while(stk.size() > t && sgn(cross(stk[stk.size() - 2], stk.back(), v[i])) <= 0) {
            stk.pop_back();
        }
        stk.push_back(v[i]);
    }
    stk.pop_back();
    return stk;
};

template<typename T>
std::vector<Point<T>> graham(std::vector<Point<T>> v) { //Graham求凸包
    Point<T> base = *min_element(v.begin(), v.end());
    std::ranges::sort(v, [&](auto p1, auto p2) ->bool {
        if(sgn(cross(base, p1, p2)) == 0) return p1 < p2;
        return sgn(cross(base, p1, p2)) == -1;
    });
    std::vector<Point<T>> stk;
    for(int i = 0; i < v.size(); ++i) {
        while(stk.size() > 1 && sgn(cross(stk[stk.size() - 2], stk.back(), v[i])) >= 0) {
            stk.pop_back();
        }
        stk.push_back(v[i]);
    }
    return stk;
}

template<typename T>
std::vector<Line<T>> halfcut(std::vector<Line<T>> lines) { //半平面交，默认左侧
    std::sort(lines.begin(), lines.end(), [&](const auto &l1, const auto &l2) ->bool {
        auto v1 = l1.t - l1.s, v2 = l2.t - l2.s;
        auto a1 = atan2(v1.y, v1.x), a2 = atan2(v2.y, v2.x);
        if(sgn(a1 - a2) != 0) return sgn(a1 - a2) == -1;
        return sgn(cross(l1.s, l1.t, l2.t)) < 0;
    });
    int l = 0, r = -1;
    std::vector<Line<T>> ls(lines.size());
    for(const auto &line : lines) {
        if(r >= l && sgn(cross(line.t - line.s, ls[r].t - ls[r].s)) == 0) continue;
        while(r > l && sgn(cross(line.s, line.t, intersection(ls[r], ls[r - 1]))) == -1) r--;
        while(r > l && sgn(cross(line.s, line.t, intersection(ls[l], ls[l + 1]))) == -1) l++;
        ls[++r] = line;
    }
    while(r > l + 1 && sgn(cross(ls[l].s, ls[l].t, intersection(ls[r], ls[r - 1]))) == -1) r--;
    return std::vector<Line<T>>(ls.begin() + l, ls.begin() + r + 1);
}

template<typename T>
std::vector<Point<T>> linesToPoints(const std::vector<Line<T>> &lines) { //直线式凸包转为点凸包
    std::vector<Point<T>> v;
    for(int i = 0; i < lines.size(); ++i) {
        v.push_back(intersection(lines[i], lines[(i + 1) % lines.size()]));
    }
    return v;
};

template<typename T>
long double area(const std::vector<Point<T>> &v) { //求凸包面积
    long double ans = 0;
    for(int i = 1; i + 1 < v.size(); ++i) {
        ans += cross(v[0], v[i], v[i + 1]);
    }
    ans /= 2;
    return ans;
}

template<typename T>
T diameter2(const std::vector<Point<T>> &v) { //旋转卡壳求凸包直径的平方
    int n = v.size();
    T res = 0;
    for(int i = 0, j = 1; i < n; ++i) {
        while(sgn(cross(v[i], v[(i + 1) % n], v[j]) - cross(v[i], v[(i + 1) % n], v[(j + 1) % n])) <= 0) {
            j = (j + 1) % n;
        }
        res = std::max({res, dis2(v[i], v[j]), dis2(v[(i + 1) % n], v[j])});
    }
```

```
    return res;
}

template<typename T>
long double diameter(const std::vector<Point<T>> &v) { //旋转卡壳求凸包直径
    return sqrtl(diameter2(v));
}

template<typename T>
long double mindistance(std::vector<Point<T>> v) { //平面最近点对O(nlog(n))
    std::sort(v.begin(), v.end());
    long double d = 1E18;
    auto cmp = [](const Point<T>& a, const Point<T>& b) {
        return a.y < b.y;
    };
    std::multiset<Point<T>, decltype(cmp)> st;
    for(int i = 0, j = 0; i < v.size(); ++i) {
        while(j < i && sgn(v[j].x - v[i].x + d) <= 0) {
            st.erase(v[j]);
            ++j;
        }
        for(auto it = st.lower_bound({v[i].x, v[i].y - T(d)}); it != st.end() && sgn
    (it->y - v[i].y - T(d + 1)) <= 0; ++it) {
            d = std::min(d, dis(v[i], *it));
        }
        st.insert(v[i]);
    }
    return d;
}

template<typename T>
long double grith(const std::vector<Point<T>> &v) { //求凸包周长
    long double ans = 0;
    for(int i = 0; i < v.size(); ++i) {
        ans += dis(v[i], v[(i + 1) % v.size()]);
    }
    return ans;
}

void solve() {

}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int T = 1;
    // std::cin >> T;
    while(T--) {
        solve();
    }
    return 0;
}
```

4 附录

4.1 VSCode 设置

VSCode 常用设置

- Auto Save (自动保存)
- Alt 键的快捷键
- Code Runner 运行快捷键
  - Run in Terminal
  - Run in Terminal
- 有时间可选: smooth

4.2 互质的规律

互质规律: 比较常见的定义 1. 较大数是质数, 两个数互质

2. 较小数是质数, 较大数不是它的倍数, 两个数互质

3. 1 与其他数互质

4. 2 与奇数互质

一些推论 1. 两个相邻的自然数一定互质

2. 两个相邻的奇数一定互质

3. n 与 2n + 1 或 2n - 1 一定互质

求差判断法如果两个数相差不大，可先求出它们的差，再看差与其中较小数是否互质。如果互质，则原来两个数一定是互质数。如：194 和 201，先求出它们的差，201 - 194 = 7，因 7 和 194 互质，则 194 和 201 是互质数。相反也成立, 对较大数也成立

求商判断法用大数除以小数，如果除得的余数与其中较小数互质，则原来两个数是互质数。如：317 和 52，317÷52 = 6.....5，因余数 5 与 52 互质，则 317 和 52 是互质数。

4.3 常数表

n	$\log_{10} n$	$n!$	$C(n, n/2)$	LCM$(1...n)$	P_n
2	0.30102999	2	2	2	2
3	0.47712125	6	3	6	3
4	0.60205999	24	6	12	5
5	0.69897000	120	10	60	7
6	0.77815125	720	20	60	11
7	0.84509804	5040	35	420	15
8	0.90308998	40320	70	840	22
9	0.95424251	362880	126	2520	30
10	1.00000000	3628800	252	2520	42
11	1.04139269	39916800	462	27720	56
12	1.07918125	479001600	924	27720	77
15	1.17609126	1.31e12	6435	360360	176
20	1.30103000	2.43e18	184756	232792560	627
25	1.39794001	1.55e25	5200300	26771144400	1958
30	1.47712125	2.65e32	155117520	1.444e14	5604
P_n	37338_{40}	204226_{50}	966467_{60}	190569292_{100}	$1e9_{114}$

$\max \omega(n)$：小于等于 n 中的数最大质因数个数
$\max d(n)$：小于等于 n 中的数最大因数个数
$\pi(n)$：小于等于 n 中的数最大互质数个数

$n \leq$	10	100	1e3	1e4	1e5	1e6
$\max \omega(n)$	2	3	4	5	6	7
$\max d(n)$	4	12	32	64	128	240
$\pi(n)$	4	25	168	1229	9592	78498
$n \leq$	1e7	1e8	1e9	1e10	1e11	1e12
$\max \omega(n)$	8	8	9	10	10	11
$\max d(n)$	448	768	1344	2304	4032	6720
$\pi(n)$	664579	5761455	5.08e7	4.55e8	4.12e9	3.7e10
$n \leq$	1e13	1e14	1e15	1e16	1e17	1e18
$\max \omega(n)$	12	12	13	13	14	15
$\max d(n)$	10752	17280	26880	41472	64512	103680
$\pi(n)$	Prime number theorem: $\pi(x) \sim \frac{x}{\log(x)}$					

4.4 常见错因

爆数据 (爆 int, 爆 longlong)
取 mod 没有取干净或者取 mod 时超范围
想不出题事算一下各种数据范围

4.5 斐波那契数列

1. $\sum_{i=1}^{n} F_i = F_{n+2} - 1$.
2. $\sum_{i=1}^{n} F_{2i-1} = F_{2n}$.
3. $\sum_{i=1}^{n} F_{2i} = F_{2n+1} - 1$.
4. $\sum_{i=1}^{n} F_i^2 = F_n F_{n+1}$.
5. $F_{n+m} = F_{n+1}F_m + F_nF_{m-1}$.
6. $F_{n-1}F_{n+1} - F_n^2 = (-1)^n$. (Cassini 恒等式)
7. $F_{2n-1} = F_n^2 + F_{n-1}^2$.
8. $F_n = \frac{F_{n+2} + F_{n-2}}{3}$.
9. $F_{2n} = F_n\big(F_{n+1} + F_{n-1}\big)$.
10. 对任意 $k \in \mathbb{N}$，有 $F_n \mid F_{nk}$.
11. 若 $F_a \mid F_b$，则 $a \mid b$.
12. $\gcd(F_n, F_m) = F_{\gcd(n,m)}$.
13. $F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$.

4.6 算法

出现的错误检查

1. 超出范围 (int, i64)
2. 取模出现错误
3. 图可能有重边或者自环

杂项 (通用)

想不到的题考虑:

1. 二分, 二分答案, 三分
2. dp
3. 离线
4. 倒推
5. 倍增

6. 建图

分块

一个二进制状态, 枚举他的所有子集状态 `for(j = status; j > 0; j = (j - 1) & status);`

中位数:

- 变成二分变成 +1 -1 判断
- 若动态维护, 每次加入一个数然后求中位数, 用对顶堆
- 如果还有删除一个数, 那么就用对顶 multiset

DP

- 背包
- 区间 dp
- DAG 图上的 dp
- 树形 dp
  - 直接 dfs
  - DFN 序:
    - 能够知道这个节点的子树大小
    - 某节点在不在此子树
    - 适用于对于一个节点向一个还没有合并过的节点合并复杂度较低的问题
  - 换根 dp
  - 基环树
- 状压 dp
- 数位 dp

搜索

- dfs, bfs
- bfs 不适用于有权图 01bfs 可适用于权值只有 0 和 1 的图
- 双向广搜: 两边各搜一半再合并

图论

- **拓扑排序:**
  - 能够处理环, 算出环的大小, 链的大小等
- **tarjan 缩点 (连通性相关)**
  - 边双联通分量: 等价于: 该子图中任意两点之间至少存在两条边互不相交的路径
  - 点双联通分量: 等价于: 任意两点之间至少存在两条内部点互不相交的路径。

树

- **倍增**
  - lca
- **树的重心**
  - 定义:
    1. 以某个节点为根时，最大子树的节点数最少，那么这个节点是重心
    2. 以某个节点为根时，每颗子树的节点数不超过总节点数的一半，那么这个节点是重心
    3. 以某个节点为根时，所有节点都走向该节点的总边数最少，那么这个节点是重心
  - 性质:
    1. 一棵树最多有两个重心，如果有两个重心，那么两个重心一定相邻
    2. 如果树上增加或者删除一个叶节点，转移后的重心最多移动一条边
    3. 如果把两棵树连起来，那么新树的重心一定在原来两棵树重心的路径上
    4. 树上的边权如果都为正数，不管边权怎么分布，所有节点都走向重心的总距离和最小
- **树的直径**
  - 求法:
    1. 两次 dfs 找两个距离最远的点不适用于有负边的树
    2. 树形 dp 对于每个点找子树中的最长的两条链适用于所有树
  - 性质:
    如果树上的边权都为正，则有如下直径相关的结论：
    1. 如果有多条直径，那么这些直径一定拥有共同的中间部分，可能是一个公共点或一段公共路径
    2. 树上任意一点，相隔最远的点的集合，直径的两端点至少有一个在其中
- 树上差分
  - 点差分
  - 边差分
- 换根 dp
- 重链剖分
- 树上启发式合并
  - 适用于对多个子树统计答案

  - 树上启发式合并的特征：
    1. 没有修改操作
    2. 可以通过遍历子树，建立信息统计，得到所有查询的答案

数学

- 数论分块
- 互质的情况
- 素数密度
- **质数判断:**
  1. 一个较小质数判断, 试除法
  2. 一个较大质数判断, miller rabin
  3. 一个范围内质数 (较多质数) 判断欧拉筛
- **质因数分解:**
  1. 数量少用试除法 $O(\sqrt{n})$
  2. 数量多欧拉筛除以 minp

字符串

- kmp
- 字符串哈希
- trie

数据结构

- 链表, 栈, 队列
- **单调栈:** 能够知道每个数前面距离最近的比它大或者小的数
- **单调队列:** 能够知道每个长度为 k 的子区间的最值
- **堆**
  - 对顶堆: 能够方便的动态维护集合中第 k 大的元素
- **并查集:**
  - 扩展域/种类并查集: 能够维护满足 1. 朋友的朋友是朋友 2. 敌人的敌人是朋友
  - 带权并查集: 维护到根的距离并且取模能够达到类似扩展域并查集的效果
- **线段树:**
  - 区间加减
  - 区间修改 Tag 设一个 ip 变量代表是否修改
  - 势能分析直接暴力修改到叶子
  - 区间合并
  - 扫描线的修改, 区间懒标记是否被选取, 因为删除的时候只会删除已经添加过的区间
  - 动态开点
- **树状数组**
  - 能够维护可差分信息
  - 能够更快的边维护边查单个点的前缀
  - 能够动态地知道每个数前面有多少个小于它的数
  - kth 知道第 k 大的数是多少
  - 树上二分
- **波纹疾走树**
  - 查询区间第 k 小的数字
  - 区间内一个数字/一段数字的频率

4.7 组合数学公式

性质 1：

$$\mathrm{C}_{\mathrm{n}}^{\mathrm{m}}=\mathrm{C}_{\mathrm{n}}^{\mathrm{n-m}}$$

性质 2：

$$\mathrm{C}_{\mathrm{n+m+1}}^{\mathrm{m}}=\sum_{i=0}^{\mathrm{m}}\mathrm{C}_{\mathrm{n+}i}^{i}$$

性质 3：

$$\mathrm{C}_{\mathrm{n}}^{\mathrm{m}}\cdot\mathrm{C}_{\mathrm{m}}^{\mathrm{r}}=\mathrm{C}_{\mathrm{n}}^{\mathrm{r}}\cdot\mathrm{C}_{\mathrm{n-r}}^{\mathrm{m-r}}$$

性质 4（二项式定理）：

$$\sum_{i=0}^{\mathrm{n}}\left(\mathrm{C}_{\mathrm{n}}^{i}\cdot x^{i}\right)=(1+x)^{\mathrm{n}}$$

$$\sum_{i=0}^{\mathrm{n}}\mathrm{C}_{\mathrm{n}}^{i}=2^{\mathrm{n}}$$

性质 5：

$$\sum_{i=0}^{\mathrm{n}}\left((-1)^{i}\cdot\mathrm{C}_{\mathrm{n}}^{i}\right)=0$$

性质 6：

$$C_n^0 + C_n^2 + \cdots = C_n^1 + C_n^3 + \cdots = 2^{n-1}$$

性质 7:

$$C_{n+m}^r = \sum_{i=0}^{\min(n,m,r)} \left(C_n^i \cdot C_m^{r-i}\right)$$

$$C_{n+m}^n = C_{n+m}^m = \sum_{i=0}^{\min(n,m)} \left(C_n^i \cdot C_m^i\right), \quad (r = n \mid r = m)$$

性质 8:

$$m \cdot C_n^m = n \cdot C_{n-1}^{m-1}$$

性质 9:

$$\sum_{i=0}^{n} \left(C_n^i \cdot i^2\right) = n(n+1) \cdot 2^{n-2}$$

性质 10:

$$\sum_{i=0}^{n} \left(C_n^i\right)^2 = C_{2n}^n$$

4.8 编译参数

-D_GLIBCXX_DEBUG : STL debugmode
-fsanitize=address : 内存错误检查
-fsanitize=undefined :UB 检查

4.9 运行脚本

Linux 运行脚本

```
#!/bin/bash
g++ -std=c++20 -O2 -Wall "$1.cpp" -o "$1" -D_GLIBCXX_DEBUG
./"$1" < in.txt > out.txt
cat out.txt
```

4.10 随机素数

979345007 986854057502126921

935359631 949054338673679153

931936021 989518940305146613

984974633 972090414870546877

984858209 956380060632801307